Theory Presentation Combinators[⋆]

Jacques Carette and Russell O'Connor [⋆⋆]

Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada

Abstract. We motivate and give semantics to *theory presentation combinators* as the foundational building blocks for a scalable library of theories. The key observation is that the *category of contexts* and fibered categories are the ideal theoretical tools for this purpose.

1 Introduction

A mechanized mathematics system, to be useful, must possess a large library of mathematical knowledge, on top of sound foundations. While sound foundations contain many interesting intellectual challenges, building a large library seems a daunting task because of its sheer volume. However, as has been well-documented [5, 6, 12], there is a tremendous amount of redundancy in existing libraries.

Our aim is to build tools that allow library developers to take advantage of all the commonalities in mathematics so as to build a large, rich library for end-users, whilst expending much less actual development effort. In other words, we continue with our approach of developing *High Level Theories* [4] through building a network of theories, by putting our previous experiments [5] on a sound theoretical basis.

1.1 The Problem

The problem which motivates this research is fairly simple: give developers of mathematical libraries the foundational tools they need to take advantage of the inherent structure of mathematical theories, as first class mathematical objects in their own right. Figure 1 shows the type of structure we are talking about: The presentation of the theory `Semigroup` strictly contains that of the theory `Magma`, and this information should not be duplicated. A further requirement is that we need to be able to selectively hide (and reveal) this structure from end-users.

[⋆] This research was supported by NSERC.

[⋆⋆] `carette@mcmaster.ca`,`roconn@mcmaster.ca`.

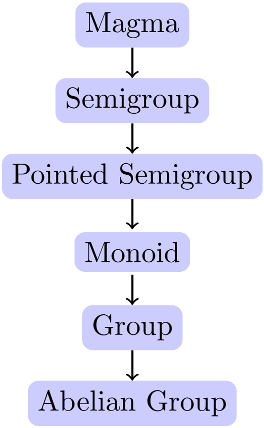

Fig. 1. Theories

The motivation for these tools should be obvious, but let us nevertheless spell it out: we simply cannot afford to spend the human resources necessary (one estimate was 140 person-years [20]; [1] explore this topic in much greater depth) to develop yet another mathematical library. In fact, as we *now* know that there is a lot of *structured* redundancy in such libraries, it would be downright foolish to not take full advantage of that. As a minor benefit, it can also help reduce errors in axiomatizations.

The motivation for being able to selectively hide or reveal some of this structure is less straightforward. It stems from our observation [4] that *in practice*, when mathematicians are *using* theories rather than developing news ones, they tend to work in a rather “flat” name space. An analogy: someone working in Group Theory will unconsciously assume the availability of all concepts from a standard textbook, with their “usual” names and meanings. As their goal is to get some work done, whatever structure system builders have decided to use to construct their system should not leak into the application domain. They may not be aware of the existence of pointed semigroups, nor should that awareness be forced upon them. Some application domains rely on the “structure of theories”, so we can allow those users to see it.

1.2 Contributions

To be explicit, our contributions include:

- A variant of the *category of contexts*, over a dependently-typed type theory as the semantics for theory presentations.
- A simple term language for building theories, using “classical” nomenclature, even though our foundations are unabashedly categorical.
- Using “tiny theories” to allow for maximal reuse and modularity.
- Taking names seriously, since these are meant for human consumption. Moreover, we further emphasize that theory presentations are purely syntactic objects, which are meant to *denote* a semantic object.
- Treating arrows seriously: while this is obvious from a categorical standpoint, it is nevertheless novel in this application.
- Giving multiple (compatible) semantics to our language, which better capture the complete knowledge context of the terms.

1.3 Plan of paper

We motivate our work with concrete examples in section 2. The theoretical foundations of our work, the fibered category of contexts, is presented in full detail in section 3. This allow us in section 4 to formalize the language of our motivation section, syntactically and semantically. We close with some discussion, related work and conclusions in sections 5–7.

2 Motivation for Theory Presentation Combinators

Let us compare the presentation of two simple theories:

```
Monoid := Theory {
  U:type;   *:(U,U) -> U;   e:U;
  axiom rightIdentity_*_e: forall x:U. x*e = x;
  axiom leftIdentity_*_e: forall x:U. e*x = x;
  axiom associative_*: forall x,y,z:U. (x*y)*z = x*(y*z)}

CommutativeMonoid := Theory {
  U:type;   *:(U,U) -> U;   e:U;
  axiom rightIdentity_*_e: forall x:U. x*e = x;
  axiom leftIdentity_*_e: forall x:U. e*x = x;
  axiom associative_*: forall x,y,z:U. (x*y)*z = x*(y*z);
  axiom commutative_*: forall x,y:U. x*y = y*x}
```

They are identical, save for the `commutative_*` axiom, as expected. Given `Monoid`, it would be much more economical to define

```
CommutativeMonoid := Monoid extended by {
  axiom commutative_*: forall x,y:U. x*y = y*x}
```

and "expand" this definition, if necessary. Of course, given `Group`, we would similarly find ourselves writing

```
CommutativeGroup := Group extended by {
  axiom commutative_*: forall x,y:U. x*y = y*x}
```

which is also wasteful, as well as dangerous: is this "the same" axiom as before, or a different one? There is no real way to tell. It is natural to further extend our language with a facility that expresses this sharing. Taking a cue from previous work, we might want to say

```
CommutativeGroup := combine CommutativeMonoid, Group over Monoid
```

Informally, this can be read as saying that `Group` and `CommutativeMonoid` are both "extensions" of `Monoid`, and `CommutativeGroup` is formed by the union (amalgamated sum) of those extensions. Another frequent feature is *renaming*: an `AbelianGroup`, while isomorphic to a `CommutativeGroup`, is usually presented additively. We could express this as

```
AbelianGroup := CommutativeGroup[ * |-> +, e |-> 0 ]
```

Unfortunately, while this “works” to build a sizeable library (say of the order of 500 concepts) in a very economical way, it is quite brittle. Let us examine the reasons. It should be clear that by `combine`, we really mean *pushout*[1]. But a pushout is a 5-ary operation on 3 objects and 2 arrows; our syntax gives the 3 objects and leaves the arrows implicit. This is a very serious mistake: these arrows are (in general) not easy to infer, especially in the presence of renaming. For example, there are two distinct arrows from `Monoid` to `Ring`, with neither arrow being “better” than the other. Furthermore, we know that pushouts can also be regarded as a 2-ary operation on arrows. In other words, even though our goal is to produce *theory presentations*, our decision to use pushouts[2] as a fundamental building block gives us no choice but to **take arrows seriously**.

So our task is now to find a category with “theory presentations” as objects, and with arrows which somehow express the notions of extending, combining and renaming as defined above. But before we explore that in depth, let us further examine our operations. First, there is nothing specific to `CommutativeGroup` in the renaming $* \mapsto +, e \mapsto 0$, this can be applied to any theory where the pairs $(*, +)$ and $(e, 0)$ have compatible signatures (including being undefined). Similarly, `extend` really defines a “construction” which can be applied whenever all the symbols used in the extension are defined. In other words, a reasonable semantics should associate a whole class of arrows[3] to these operations.

But there is one more aspect to consider: in all our examples above, we have used short, meaningful names. While great for humans, they are in part at fault in the failure of being able to infer arrows. If, like in MMT [15], we used long names, might we be able to build a robust system? Maybe so, but it would immediately fall afoul of our second requirement: irrelevant information such as choices made by developers regarding the order in which to build theories, would leak into the long names, and thus be seen by users. Furthermore, when there is ambiguity, a long name system can indeed resolve that ambiguity, but at too high a cost to humans in absurdly long names for certain concepts.

In other words, to be able to maintain human-readable names for all concepts, we will put the burden on the *library developers* to come up with a reasonable naming scheme, rather than to push that issue onto end users. Another way to see this is that symbol choice carries a lot of

[1] Following Burstall and Goguen [2] and Smith [17, 18] and many others since.

[2] which will in fact become pullbacks

[3] We are being deliberately vague here, Section 3 will make this precise.

intentional, as well as contextual, information which is commonly used in mathematical practice. Thus, to avoid leaking irrelevant information and to maintain intentional/contextual information, we will insist that on **taking names seriously**.

3 Category of contexts

We observe that theories from the previous section can all be specified as contexts of some dependent type theory. The work in this paper is abstract over the exact details of the dependent type theory,[4] so we simply assume that some dependent type theory is given. Following Cartmell [7], we form the category of contexts $\mathbb{C}$ of the given dependent type theory. The objects of $\mathbb{C}$ are contexts Γ that occur in judgements like $\Gamma \vdash s : \sigma$ of the dependent type theory. A context Γ consists of a sequence of pairs of labels and types (or kinds or propositions),

$$\Gamma := \langle x_0 : \sigma_0; \ldots; x_{n-1} : \sigma_{n-1} \rangle \,,$$

such that for each $i < n$ the judgement

$$\langle x_0 : \sigma_0; \ldots; x_{i-1} : \sigma_{i-1} \rangle \vdash \sigma_i : \text{Type}$$

holds (resp. : Kind, or : Prop). Contexts of dependent type theory can be used to define the types, operations, relations and axioms of a theory. We will use the abbreviation $\langle x : \sigma \rangle_0^{n-1}$ for a context Γ, and $\fatsemi$ for concatenation of two such sequences.

Example 1. We can define the theory of semigroups via

$$\texttt{Semigroup} := \left\langle \begin{array}{c} U : \text{Type} \\ (*) : U \times U \to U \\ \text{associative} : \forall x, y, z : U.\, (x * y) * z = x * (y * z) \end{array} \right\rangle$$

where we use Haskell-style notation where $(\Box)$ indicates (the name of) a binary function used infix in terms.

Normally contexts are considered up to α-equivalence, that is, renaming or permuting the labels of a context makes no difference. But since labels do make a difference, we will not do so. However, α-equivalent *terms* and *types* continue to be considered equivalent.

[4] In fact, we expect this work to apply not only to dependent type theories, but to any classifying category [13].

Example 2. The signature for `AdditiveSemigroup` is given as the context

$$\left\langle \begin{aligned} U &: \mathrm{Type} \\ (+) &: U \times U \to U \\ \mathrm{associative} &: \forall x, y, z : U.\, (x + y) + z = x + (y + z) \end{aligned} \right\rangle$$

Traditionally `Semigroup` and `AdditiveSemigroup` would be considered the same context because they are α-equivalent.

In the rest of this section, we will use the convention that $\Gamma = \langle x : \sigma \rangle_0^{n-1}$ and $\Delta = \langle y : \tau \rangle_0^{m-1}$. Given two contexts Γ and Δ, a morphism $\Gamma \to \Delta$ of $\mathbb{C}$ consists of an assignment $[y_0 \mapsto t_0, \ldots, y_m \mapsto t_{m-1}]$, abbreviated as $[y \mapsto t]_0^{m-1}$ where the $t_0, \ldots, t_{m-1}$ are terms such that

$$\Gamma \vdash t_0 : \tau_0 \qquad \ldots \qquad \Gamma \vdash t_{m-1} : \tau_{m-1} \, [y \mapsto t]_0^{m-2}$$

all hold, where $\tau \, [y \mapsto t]_0^{i}$ denotes the type τ with the labels $y_0, \ldots, y_i$ substituted by the corresponding terms of the assignment. We will also use $\bowtie$ to denote concatenation of assignments, and $\left[y_{f(j)} \mapsto t_{g(j)}\right]_{j=a}^{b}$ for the "obvious" generalized assignment.

Notice that an arrow from Γ to Δ is an assignment from the labels of Δ to terms in Γ. This definition of an arrow may seem backwards at first, but it is defined this way because arrows transform "models" of theories of Γ to "models" of theories of Δ. For example, every Abelian Semigroup is, or rather can be transformed into, an Additive Semigroup by simply forgetting that the Semigroup is Abelian. A later example 4 will give the explicit arrow from Abelian Semigroup to Additive Semigroup that captures this transformation.

Let us fix $\mathbb{V}$ as the (countable) infinite set of labels used in contexts. If $\pi : \mathbb{V} \to \mathbb{V}$ is a permutation of labels, then we can define an action of this permutation on terms, types and contexts:

$$\pi \cdot \langle x : \sigma \rangle_0^{n-1} := \langle \pi\,(x_0) : \pi \cdot \sigma_0; \ldots; \pi\,(x_{n-1}) : \pi \cdot \sigma_{n-1} \rangle \equiv \langle \pi x : \pi \cdot \sigma \rangle_0^{n-1}$$

where $\pi \cdot \sigma_i$ is the action induced on the (dependent) type σ_i by renaming labels. The action of π induces an endofunctor $(\pi \cdot -) : \mathbb{C} \to \mathbb{C}$. Furthermore, each permutation $\pi : \mathbb{V} \to \mathbb{V}$ induces a natural transformation in $I_\pi : (\pi \cdot -) \Rightarrow \mathrm{id}_{\mathbb{C}}$ where

$$I_\pi\,(\Gamma) := [x_0 \mapsto \pi\,(x_0)\,, \ldots, x_{n-1} \mapsto \pi\,(x_{n-1})] : \pi \cdot \Gamma \to \Gamma.$$

We call an assignment of the form $I_\pi\,(\Gamma)$ a **renaming**. Because permutations are invertible, each renaming $I_\pi\,(\Gamma) : \pi \cdot \Gamma \to \Gamma$ is an isomorphism whose inverse is the renaming $I_{\pi^{-1}}\,(\pi \cdot \Gamma) : \Gamma \to \pi \cdot \Gamma$. From this we can see that α-equivalent contexts are isomorphic.

Example 3. Let $\pi : \mathbb{V} \to \mathbb{V}$ be some permutation such that $\pi(U) = U$, $\pi((*)) = (+)$, and $\pi(\text{associative}) = \text{associative}$. By the definition of $I_\pi(\texttt{Semigroup}) : \texttt{AdditiveSemigroup} \to \texttt{Semigroup}$, we have that

$$I_\pi(\texttt{Semigroup}) := [U \mapsto U; (*) \mapsto (+); \text{associative} \mapsto \text{associative}]$$

is a renaming isomorphism between the contexts in examples 1 and 2.

The category of nominal assignments, $\mathbb{B}$, a sub-category of $\mathbb{C}$ will be quite important for use. For example, theorem 2 will show that $\mathbb{B}$ is the base category of a fibration.

Definition 1. *The category of nominal assignments, $\mathbb{B}$, has the same objects as $\mathbb{C}$, but only those morphisms whose terms are labels.*

Thus a morphism in $\mathbb{B}$ is an assignment of the form $\left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-1}$ such that the judgements

$$\Gamma \vdash x_{a(0)} : \tau_0 \qquad \ldots \qquad \Gamma \vdash x_{a(m-1)} : \tau_{m-1} \left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-2}$$

all hold.

Definition 2. *We define Γ to be a **sub-context** of Γ^+ if every element $x : \tau$ of Γ occurs in Γ^+.*

Definition 3. *We call an assignment $A : \Gamma \to \Delta$ a diagonal assignment if A is of the form $[y \mapsto y]_0^{n-1}$ (where $\Delta = \langle y : \tau \rangle_0^{n-1}$), denoted by $\delta_\Delta : \Gamma \to \Delta$.*

Definition 4. *An assignment $A : \Gamma^+ \to \Gamma$ is an* extension *when Γ is a sub-context of Γ^+, and A is the diagonal assignment.*

Notice that an extension points from the extended context to the sub-context. This is the reverse from what Burstall and Goguen [2] use (and most of the algebraic specification community followed their lead). Our direction is inherited from $\mathbb{C}$, the category of contexts, which is later required by theorem 2 to satisfy the technical definition of a fibration.

Example 4. Consider the theory $\texttt{AbelianSemigroup}$ given as

$$\left\langle \begin{array}{r} U : U : \text{Type} \\ (+) : U \times U \to U \\ \text{associative} : \forall x, y, z : U.\, (x + y) + z = x + (y + z) \\ \text{commutative} : \forall x, y : U.\, (x + y) = (y + x) \end{array} \right\rangle$$

Then $\delta_{\texttt{AdditiveSemigroup}} : \texttt{AbelianSemigroup} \to \texttt{AdditiveSemigroup}$ is an extension.

Example 5. Consider the following two distinct contexts $(\mathtt{C}_1, \mathtt{C}_2)$ for the theory of left unital Magmas with the order of their operators swapped:

$$\left\langle \begin{aligned} U &: \text{Type} \\ e &: U \\ (*) &: U \times U \to U \\ \text{leftIdentity} &: \forall x : U . e * x = x \end{aligned} \right\rangle \quad \left\langle \begin{aligned} U &: \text{Type} \\ (*) &: U \times U \to U \\ e &: U \\ \text{leftIdentity} &: \forall x : U . e * x = x \end{aligned} \right\rangle$$

The diagonal assignment $\delta_{C_1} : \mathtt{C}_2 \to \mathtt{C}_1$ is an extension (as is $\delta_{C_2} : \mathtt{C}_1 \to \mathtt{C}_2$).

Notice that, for any given contexts Γ^+ and Γ, there exists an extension $\Gamma^+ \to \Gamma$ if and only if Γ is a sub-context of Γ^+. If Γ is a sub-context of Γ^+ then the diagonal assignment $\delta_\Gamma : \Gamma^+ \to \Gamma$ is the unique extension.

In general, a renaming $I_\pi : \pi \cdot \Gamma \to \Gamma$ will not be an extension unless π is the identity on the labels from Γ. In our work, both renaming and extentions are used together, so we want to consider a broader class of nominal assignments that include both extensions and renamings.

Definition 5. *Those nominal assignments where every label occurs at most once will be called general extensions.*

We see that for every permutation of labels $\pi : \mathbb{V} \to \mathbb{V}$ and every context Γ that $I_\pi(\Gamma) : \pi \cdot \Gamma \to \Gamma$ is a general extension (and hence also a nominal assignment).

Theorem 1. *Every general extension $A : \Gamma^+ \to \Delta$ can be turned into an extension by composing it with an appropriate renaming.*

The proof of this theorem, along with all other theorems, lemmas and corollaries in this section can be found in Appendix A.

Corollary 1. *Every general extension $A : \Gamma^+ \to \Delta$ can be decomposed into an extension $A_e : \Gamma^+ \to \Gamma$ followed by a renaming $A_r : \Gamma \to \Delta$.*

These general extensions form a category which plays an important rôle.

$$\begin{tikzcd} \Gamma^+ \arrow[r, "f^+"] \arrow[d, "A"'] & \Delta^+ \arrow[d, "B"] \\ \Gamma \arrow[r, "f^-"'] & \Delta \end{tikzcd}$$

Definition 6. *The category of general extensions $\mathbb{E}$ has all general extensions from $\mathbb{B}$ as objects, and given two general extensions $A : \Gamma^+ \to \Gamma$ and $B : \Delta^+ \to \Delta$, an arrow $f : A \to B$ is a commutative square from $\mathbb{B}$. We will denote this commutative square by $\langle f^+, f^- \rangle : A \to B$.*

We remind the reader of the usual convention in category theory where arrows include their domain and codomain as part of their structure (which we implicitly use in the definition above).

Lemma 1. *Every general extension is isomorphic in $\mathbb{E}$ to an extension $B : \mathit{\Gamma}^{\circ} \to \mathit{\Gamma}$ where $\mathit{\Gamma}$ is an initial segment of $\mathit{\Gamma}^{\circ}$.*

This category of general extensions $\mathbb{E}$ is fibered over the category $\mathbb{B}$ by the codomain functor $\mathrm{cod} : \mathbb{E} \to \mathbb{B}$. Given general extensions $A : \mathit{\Gamma}^{+} \to \mathit{\Gamma}$ and $B : \mathit{\Delta}^{+} \to \mathit{\Delta}$ and a morphism $\langle f^{+}, f^{-} \rangle : A \to B$ in $\mathbb{E}$ we have

$$\mathrm{cod}\,(A) := \mathit{\Gamma} \qquad\qquad \mathrm{cod}\,(f) := f^{-}$$

Theorem 2. *The functor* $\mathrm{cod} : \mathbb{E} \to \mathbb{B}$ *is a fibration.*

Corollary 2. *Given $u : \mathit{\Gamma} \to \mathit{\Delta}$, a general extension $A : \mathit{\Delta}^{+} \to \mathit{\Delta}$, and a cartesian lifting $\bar{u}\,(A) : u^{*}\,(A) \to A$, if u is a general extension, then $\bar{u}\,(A)^{+}$ is also a general extension.*

Example 6. The nominal assignment (and general extension)

$$u := \begin{bmatrix} U \mapsto U \\ (*) \mapsto (+) \\ \text{associative} \mapsto \text{associative} \end{bmatrix} : \texttt{AbelianSemigroup} \to \texttt{Semigroup}$$

and the extension $A := \delta_{\texttt{Semigroup}} : \texttt{Monoid} \to \texttt{Semigroup}$ induce the existence (via theorem 2) of some Cartesian lifting $\bar{u}\,(A) : u^{*}\,(A) \to A$ in $\mathbb{E}$. One example of such a Cartesian lifting for $\bar{u}$ is

$$\begin{tikzcd}
\texttt{AbelianMonoid} \arrow[r, "\bar{u}(A)^{+}"'] \arrow[d, "u^{*}(A)"'] & \texttt{Monoid} \arrow[d, "A"] \\
\texttt{AbelianSemigroup} \arrow[r, "u"] & \texttt{Semigroup}
\end{tikzcd}$$

where $\texttt{AbelianMonoid}$ is

$$\left\langle \begin{aligned} U &: \text{Type} \\ 0 &: U \\ (+) &: U \times U \to U \\ \text{rightIdentity} &: \forall x : U . x + 0 = x \\ \text{leftIdentity} &: \forall x : U . 0 + x = x \\ \text{associative} &: \forall x, y, z : U .\, (x + y) + z = x + (y + z) \\ \text{commutative} &: \forall x, y : U .\, (x + y) = (y + x) \end{aligned} \right\rangle$$

and $u^*(A)$: `AbelianMonoid` → `AbelianSemigroup` is the diagonal assignment, where $\bar{u}(A)^+$: `AbelianMonoid` → `Monoid` is

$$\bar{u}(A)^+ := \begin{bmatrix} U \mapsto U \\ e \mapsto 0 \\ (*) \mapsto (+) \\ \text{rightIdentity} \mapsto \text{rightIdentity} \\ \text{leftIdentity} \mapsto \text{leftIdentity} \\ \text{associative} \mapsto \text{associative} \end{bmatrix}$$

$$\begin{tikzcd}
\langle U : \mathit{Type} \rangle \arrow[dr, dashed, "f"] \arrow[drr, bend left, "id"] \arrow[ddr, bend right, "id"'] & & \\
& \left\langle \begin{array}{c} U : \mathit{Type} \\ U' : \mathit{Type} \end{array} \right\rangle \arrow[r, "\bar{u}(u)^+"] \arrow[d, "u^*(u)"] & \langle U : \mathit{Type} \rangle \arrow[d, "u"] \\
& \langle U : \mathit{Type} \rangle \arrow[r, "u"'] & \langle\rangle
\end{tikzcd}$$

In almost all of the development of the algebraic hierarchy, the nominal assignments that we use are all general extensions. However, it is important to note that the definition of a Cartesian lifting requires nominal assignments that are not necessarily general extensions, even if all the inputs are general extensions.

Consider the simple case (pictured above) where $u : \langle U : \text{Type} \rangle \to \langle\rangle$ is the unique extension, and a Cartesian lifting of u over itself. The mediating arrow for $\text{id} : \langle U : \text{Type} \rangle \to \langle U : \text{Type} \rangle$ and itself must be

$$\begin{aligned} f &: \langle U : \text{Type} \rangle \to \left\langle U : \text{Type}; U' : \text{Type} \right\rangle \\ f &:= \left[U \mapsto U, U' \mapsto U \right] \end{aligned}$$

which is not a general extension.

4 Semantics of Theory Presentation Combinators

Like in the previous section, we will assume that we have a background type theory with well-formedness *judgments*, which defines four different sorts, namely ($\mathsf{Type}, \mathsf{Term}, \mathsf{Kind}, \mathsf{Prop}$). The symbols used in the type theory itself will be called *labels*, whereas the symbols used for theory presentations will be called *names*. As above, $a \mapsto b$ denotes a *substitution*. Using this, we can define the formal syntax for our combinators as follows.

$$\begin{array}{rl} a, b, c \in \text{labels} & \tau \in \mathsf{Type} \\ A, B, C \in \text{names} & k \in \mathsf{Kind} \\ l \in \text{judgments}^* & t \in \mathsf{Term} \\ r \in (a_i \mapsto b_i)^* & \theta \in \mathsf{Prop} \end{array} \qquad \begin{array}{rl} \mathsf{tpc} ::= & \mathsf{extend}\ A\ \mathsf{by}\ \{l\} \\ & \mid \mathsf{combine}\ A\ r_1,\ B\ r_2 \\ & \mid A\ ;\ B \\ & \mid A\ r \\ & \mid \mathsf{Empty} \\ & \mid \mathsf{Theory}\ \{l\} \end{array}$$

Intuitively, the six forms correspond to: extending a theory with new knowledge, combining two theories into a larger one, sequential composition of theories, renaming, a constant for the Empty theory, and an explicit theory.

What we do next is slightly unusual: rather than give a single denotational semantics, we will give *two*, one in terms of objects of $\mathbb{B}$, and one in terms of objects of $\mathbb{E}$ (which are special arrows in $\mathbb{B}$). In fact, we have a third semantics, in terms of (partial) Functors over the contextual category, but we will omit it for lack of space. First, we give the semantics in terms of objects of $\mathbb{B}$, where $[\![-]\!]_\pi$ is the (obvious) semantics in $\mathbb{V} \to \mathbb{V}$ of a renaming.

$$\begin{aligned} [\![-]\!]_{\mathbb{B}} : \mathsf{tpc} &\rightharpoonup |\mathbb{B}| \\ [\![\mathsf{Empty}]\!]_{\mathbb{B}} &= \langle\,\rangle \\ [\![\mathsf{Theory}\ \{l\}]\!]_{\mathbb{B}} &\cong \langle l \rangle \\ [\![A\ r]\!]_{\mathbb{B}} &= [\![r]\!]_\pi \cdot [\![A]\!]_{\mathbb{B}} \\ [\![A; B]\!]_{\mathbb{B}} &= [\![B]\!]_{\mathbb{B}} \\ [\![\mathsf{extend}\ A\ \mathsf{by}\ \{l\}]\!]_{\mathbb{B}} &\cong [\![A]\!]_{\mathbb{B}} \fatsemi \langle l \rangle \\ [\![\mathsf{combine}\ A_1 r_1, A_2 r_2]\!]_{\mathbb{B}} &\cong D \end{aligned}$$

$$\begin{tikzcd} D \arrow[r, "{[\![r_1]\!]_\pi \circ \delta_{A_1}}"] \arrow[d, "{[\![r_2]\!]_\pi \circ \delta_{A_2}}"] & A_1 \arrow[d, "\delta_A"] \\ A_2 \arrow[r, "\delta_A"'] & A \end{tikzcd}$$

where D comes from the (potential) pullback diagram on the right, in which we omit $[\![-]\!]_{\mathbb{B}}$ around the As for clarity. We use $\cong$ to abbreviate "when the rhs is a well-formed context". For the semantics of combine, it must be the case where the diagram at right is a *pullback* (in $\mathbb{B}$), where A is the greatest lower bound context $[\![A_1]\!]_{\mathbb{B}} \sqcap [\![A_2]\!]_{\mathbb{B}}$. Furthermore $[\![r_1]\!]_\pi$ and $[\![r_2]\!]_\pi$ must leave A invariant. We remind the reader of the requirement for these renamings: the users must pick which cartesian lifting they want, and this cannot be done automatically (as demonstrated at the end of last section).

The second semantics, is in terms of the objects of $\mathbb{E}$, in other words, the *special* arrows of $\mathbb{B}$, as defined in Section 3.

$$\begin{aligned}
\llbracket - \rrbracket_{\mathbb{E}} :\ & \mathsf{tpc} \rightharpoonup |\mathbb{E}| \\
\llbracket \mathsf{Empty} \rrbracket_{\mathbb{E}} =\ & \mathsf{id}_{\langle\,\rangle} \\
\llbracket \mathsf{Theory}\ \{l\} \rrbracket_{\mathbb{E}} \cong\ & !_{\langle l \rangle} \\
\llbracket A\ r \rrbracket_{\mathbb{E}} =\ & \llbracket r \rrbracket_{\pi} \cdot \llbracket A \rrbracket_{\mathbb{E}} \\
\llbracket A; B \rrbracket_{\mathbb{E}} =\ & \llbracket A \rrbracket_{\mathbb{E}} \circ \llbracket B \rrbracket_{\mathbb{E}} \\
\llbracket \mathsf{extend}\ A\ \mathsf{by}\ \{l\} \rrbracket_{\mathbb{E}} \cong\ & \delta_A \\
\llbracket \mathsf{combine}\ A_1 r_1, A_2 r_2 \rrbracket_{\mathbb{E}} \cong\ & \llbracket r_1 \rrbracket_{\pi} \circ \delta_{T_1} \circ \llbracket A_1 \rrbracket_{\mathbb{E}} \\
\cong\ & \llbracket r_2 \rrbracket_{\pi} \circ \delta_{T_2} \circ \llbracket A_2 \rrbracket_{\mathbb{E}}
\end{aligned}$$

$$\begin{tikzcd}
D \arrow[r, "\llbracket r_1 \rrbracket_{\pi} \circ \delta_{T_1}"] \arrow[d, "\llbracket r_2 \rrbracket_{\pi} \circ \delta_{T_2}"] & T_1 \arrow[d, "A_1"] \\
T_2 \arrow[r, "A_2"'] & T
\end{tikzcd}$$

The diagram on the right has to be verified to be a pullback diagram (this is why the semantics is partial here too). Here we assume $\llbracket A_1 \rrbracket_{\mathbb{E}} \in \mathit{Hom}(T_1, T)$ and $\llbracket A_2 \rrbracket_{\mathbb{E}} \in \mathit{Hom}(T_2, T)$, and that both $\llbracket r_1 \rrbracket_{\pi}$ and $\llbracket r_2 \rrbracket_{\pi}$ leave T invariant.

Theorem 3. *For all* tpc *terms except* combine, $\llbracket s \rrbracket_{\mathbb{B}} = \mathsf{dom}\llbracket s \rrbracket_{\mathbb{E}}$. *When* $s \equiv \mathsf{combine}\ A_1 r_1, A_2 r_2$*, if* $\mathsf{cod}\,(\llbracket A_1 \rrbracket_{\mathbb{E}}) = \mathsf{cod}\,(\llbracket A_2 \rrbracket_{\mathbb{E}}) = \llbracket A_1 \rrbracket_{\mathbb{B}} \sqcap \llbracket A_2 \rrbracket_{\mathbb{B}}$*, and neither arrows* $\llbracket A_1 \rrbracket_{\mathbb{E}}$ *nor* $\llbracket A_2 \rrbracket_{\mathbb{E}}$ *involve renamings, then* $\llbracket s \rrbracket_{\mathbb{B}} = \mathsf{dom}\llbracket s \rrbracket_{\mathbb{E}}$ *in that case as well.*

The proof is a straightforward comparison of the semantic equations. This theorem basically says that, as long as we only use combine on the "natural" base of two arrows which are pure extensions, our semantics are compatible. In a *tiny theories* setting, this can be arranged.

5 Discussion

It is important to note that we are essentially parametric in the underlying type theory. This should allow us to be able to generalize our work in ways similar to Kohlhase and Rabe's MMT [15].

The careful reader might have notice that in the syntax of section 2, our `combine` had an `over` keyword. This allowed our previous implementation [5] to come partway to the $\mathbb{E}$ semantics above. This is a straightforward extension to the semantics: $\llbracket \mathsf{combine}\ A_1 r_1, A_2 r_2\ \mathsf{over}\ C \rrbracket_{\mathbb{B}}$ would replace $A = \llbracket A_1 \rrbracket_{\mathbb{B}} \sqcap \llbracket A_2 \rrbracket_{\mathbb{B}}$ with $\llbracket C \rrbracket_{\mathbb{B}}$, with corresponding adjustments to the rest of the pushout diagram. For $\llbracket - \rrbracket_{\mathbb{E}}$, one would insist that $\mathsf{cod}\,(\llbracket A_1 \rrbracket_{\mathbb{E}}) = \mathsf{cod}\,(\llbracket A_2 \rrbracket_{\mathbb{E}}) = \llbracket C \rrbracket_{\mathbb{B}}$.

What is more promising[5] still is that most of our terms can also be interpreted as *Functors* between fibered categories. This gives us a semantic for each term as a "construction", which can be reused (as in our example with commutativity in section 2). Furthermore, since fibered categories interact well will limits and colimits, we should also be able to combine constructions and diagrams so as to fruitfully capture further structure in theory hierarchies.

It should also be noted that our work also extends without difficulty to having *definitions* (and other such conservative extensions) in our contexts. This is especially useful when transporting theorems from one setting to another, as is done when using the "Little Theories" method [9]. We also expect our work to extend to allow Cartesian liftings of extensions over arbitrary assignments (aka views) from the full category of contexts.

Lastly, we have implemented a "flattener" for our semantics, which basically turns a presentation A into a flat presentation $\mathsf{Theory}\{\ l\}$ by computing $\mathsf{cod}\,(\llbracket A \rrbracket_{\mathbb{E}})$. This fulfils our second requirement, where the method of construction of a theory is invisible to users of flat theories.

6 Related Work

We will not consider work in universal algebra, institutions or categorical logic as "related", since they employ α-equivalence on labels (as well as on bound variables), which we consider un-helpful for theory presentations meant for human consumption. We also leave aside much interesting work on dependent record types (which we use), as these are but one implementation method for theories, and we consider *contexts* as a much more fundamental object.

We have been highly influenced by the early work of Burstall and Goguen [2, 3], and Doug Smith's Specware [17, 18]. They gave us the basic semantic tools we needed. But we quickly found out, much to our dismay, that neither of these approaches seemed to scale very well. Later, we were hopeful that CASL [8] might work for us, but then found that their own base library was improperly factored and full of redundancies. Of the vast algebraic specification literature around this topic, we want to single out the work of Oriat [14] on isomorphism of specification graphs as capturing similar ideas to ours on extreme modularity. And it cannot be emphasized enough how crucial Bart Jacob's book [13] has been to our work.

From the mathematical knowledge management side, it should be clear that MMT [15] is closely related. The main differences are that they

[5] Work in progress

are quite explicit about being foundations-independent (it is implicit in our work), they use long names, and their theory operations are mostly theory-internal, while ours are external. This makes a big difference, as it allows us to have multiple semantics, while theirs has to be fixed. And, of course, the work presented in the current paper covers just a small part of the vast scope of MMT.

There are many published techniques and implementations of algebraic hierarchies in dependently typed proof assistants including [11, 19, 10, 16]. Our work does not compete with these implementations, but rather complements them. More specifically, we envision our work as a meta-language which can be used to specify algebraic hierarchies, which can subsequently be implemented by using any of the aforementioned techniques. In particular we note that maintaining the correct structures for packed-classes of [10] is particularly difficult, and deriving the required structures from a hierarchy specification would alleviate much of this burden. Other cited work, (for example [16]) focus on other difficult problems such as *usability*, via providing coercions and unification hints to match particular terms to theories. Even though some similar techniques (categorical pullbacks) are used in a similar context, the details are very different.

7 Conclusion

There has been a lot of work done in mathematics to give structure to mathematical theories, first via universal algebra, then via category theory (e.g. Lawvere theories). But even though a lot of this work started out being somewhat syntactic, very quickly it became mostly semantic, and thus largely useless for the purposes of concrete implementations.

We make the observation that, with a rich enough type theory, we can identify the category of theory presentations with the opposite of the category of contexts. This allows us to draw freely from developments in categorical logic, as well as to continue to be inspired by algebraic specifications. Interestingly, key here is to make the opposite choice as Goguen's in two ways: our base language is firmly higher-order, while our "module" language is first-order, and we work in the opposite category.

We provide a simple-to-understand term language of "theory expression combinators", along with multiple (categorical) semantics. We have shown that these fit our requirements of allowing to capture mathematical structure, while also allowing this structure to be hidden from users.

Even more promising, our use of very standard categorical constructions points the way to simple generalizations which should allow us to capture even more structure, without having to rewrite our library. Furthermore, as we are independent of the details of the type theory, this structure seems very robust, and our combinators should thus port easily to other systems.

References

1. Asperti, A., Sacerdoti Coen, C.: Some considerations on the usability of interactive provers. In: Proceedings of the 10th ASIC and 9th MKM international conference, and 17th Calculemus conference on Intelligent computer mathematics. pp. 147–156. AISC'10/MKM'10/Calculemus'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1894483.1894498
2. Burstall, R.M., Goguen, J.A.: Putting theories together to make specifications. In: IJCAI. pp. 1045–1058 (1977)
3. Burstall, R.M., Goguen, J.A.: The semantics of clear, a specification language. In: Bjørner, D. (ed.) Abstract Software Specifications. Lecture Notes in Computer Science, vol. 86, pp. 292–332. Springer (1979)
4. Carette, J., Farmer, W.M.: High-level theories. In: Autexier, A.e.a. (ed.) Intelligent Computer Mathematics. Lecture Notes in Computer Science, vol. 5144, pp. 232–245. Springer-Verlag (2008)
5. Carette, J., Farmer, W.M., Jeremic, F., Maccio, V., O'Connor, R., Tran, Q.: The mathscheme library: Some preliminary experiments. Tech. rep., University of Bologna, Italy (2011), uBLCS-2011-04
6. Carette, J., Kiselyov, O.: Multi-stage programming with functors and monads: Eliminating abstraction overhead from generic code. Sci. Comput. Program. 76(5), 349–375 (2011)
7. Cartmell, J.: Generalised algebraic theories and contextual categories. Annals of Pure and Applied Logic 32, 209 – 243 (1986), http://www.sciencedirect.com/science/article/pii/0168007286900539
8. CoFI (The Common Framework Initiative): Casl Reference Manual. LNCS Vol. 2960 (IFIP Series), Springer-Verlag (2004)
9. Farmer, W.M., Guttman, J.D., Thayer, F.J.: Little theories. In: CADE-11: Proceedings of the 11th International Conference on Automated Deduction. pp. 567–581. Springer-Verlag, London, UK (1992)
10. Garillot, F., Gonthier, G., Mahboubi, A., Rideau, L.: Packaging mathematical structures. In: Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics. pp. 327–342. TPHOLs '09, Springer-Verlag, Berlin, Heidelberg (2009), \url{http://dx.doi.org/10.1007/978-3-642-03359-9_23}
11. Geuvers, H., Wiedijk, F., Zwanenburg, J.: A constructive proof of the fundamental theorem of algebra without using the rationals. In: TYPES '00: Selected papers from the International Workshop on Types for Proofs and Programs. pp. 96–111. Springer-Verlag, London, UK (2002)
12. Grabowski, A., Schwarzweller, C.: On duplication in mathematical repositories. In: Autexier, S., Calmet, J., Delahaye, D., Ion, P., Rideau, L., Rioboo, R., Sexton, A. (eds.) Intelligent Computer Mathematics, Lecture Notes in Computer Science, vol. 6167, pp. 300–314. Springer Berlin / Heidelberg (2010)

13. Jacobs, B.: Categorical Logic and Type Theory. No. 141 in Studies in Logic and the Foundations of Mathematics, North Holland, Amsterdam (1999)
14. Oriat, C.: Detecting equivalence of modular specifications with categorical diagrams. Theor. Comput. Sci. 247(1-2), 141–190 (2000)
15. Rabe, F., Kohlhase, M.: A Scalable Module System, available from `http://kwarc.info/frabe/Research/mmt.pdf`
16. Sacerdoti Coen, C., Tassi, E.: Nonuniform coercions via unification hints. In: Hirschowitz, T. (ed.) TYPES. EPTCS, vol. 53, pp. 16–29 (2009)
17. Smith, D.R.: Constructing specification morphisms. Journal of Symbolic Computation 15, 5–6 (1993)
18. Smith, D.R.: Mechanizing the development of software. In: Broy, M., Steinbrueggen, R. (eds.) Calculational System Design, Proceedings of the NATO Advanced Study Institute, pp. 251–292. IOS Press, Amsterdam (1999)
19. Spitters, B., van der Weegen, E.: Type classes for mathematics in type theory. Mathematical Structures in Computer Science 21(4), 795–825 (2011)
20. Wiedijk, F.: Estimating the cost of a standard library for a mathematical proof checker (2001), `\url{http://www.cs.ru.nl/~freek/notes/mathstdlib2.pdf}`

A Proofs

Given a nominal assignment of the form $\left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-1}$, we call the function $a : \mathbb{N}_{|\Delta|} \to \mathbb{N}_{|\Gamma|}$ above an **indexing function**. The indexing function of a nominal assignment is sufficient to specify the nominal assignment. Given a morphism $A : \Gamma \to \Delta$ with indexing function $a : \mathbb{N}_{|\Delta|} \to \mathbb{N}_{|\Gamma|}$ and a morphism $B : \Delta \to \Xi$ with indexing function $b : \mathbb{N}_{|\Xi|} \to \mathbb{N}_{|\Delta|}$, that the composition $B \circ A : \Gamma \to \Xi$ has $a \circ b : \mathbb{N}_{|\Xi|} \to \mathbb{N}_{|\Gamma|}$ for its indexing function.

Proof (of Theorem 1, Section 3). Suppose $A = \left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-1}$. Because a is injective for general extensions, we can select a permutation of names $\pi_A : \mathbb{V} \to \mathbb{V}$ such that $\pi_A\left(y_i\right) = x_{a(i)}$. This implies that $\pi_A \cdot \Delta$ is a subcontext of Γ^+. Consider the renaming $I_{\pi_A^{-1}}\left(\pi_A \cdot \Delta\right) : \Delta \to \pi_A \cdot \Delta$. We see that the composition $I_{\pi_A^{-1}}\left(\pi_A \cdot \Delta\right) \circ A : \Gamma^+ \to \pi_A \cdot \Delta$ is an extension since

$$\begin{aligned} I_{\pi_A^{-1}}\left(\pi_A \cdot \Delta\right) \circ A &= \left[\pi_A\left(y_i\right) \mapsto y_i\right]_{i=0}^{m-1} \circ \left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-1} \\ &= \left[\pi_A\left(y_i\right) \mapsto x_{a(i)}\right]_{i=0}^{m-1} = \left[x_{a(i)} \mapsto x_{a(i)}\right]_{i=0}^{m-1} \end{aligned}$$

is the diagonal assignment. □

Proof (of Corollary 1, Section 3). From the previous theorem we know that there is a permutation $\pi_A : \mathbb{V} \to \mathbb{V}$ such that $I_{\pi_A^{-1}}\left(\pi_A \cdot \Delta\right) \circ A :$

$\mathit{\Gamma}^+ \to \pi_A \cdot \mathit{\Delta}$ is an extension. Let $\mathit{\Gamma} := \pi_A \cdot \mathit{\Delta}$ and let $A_e := I_{\pi_A^{-1}} (\pi_A \cdot \mathit{\Delta}) \circ A$. Then we can take $A_r := I_{\pi_A} (\mathit{\Delta}) : \mathit{\Gamma} \to \mathit{\Delta}$. We immediately see that

$$\begin{aligned} A_r \circ A_e &= I_{\pi_A} (\mathit{\Delta}) \circ I_{\pi_A^{-1}} (\pi_A \cdot \mathit{\Delta}) \circ A \\ &= I_{\pi_A} (\mathit{\Delta}) \circ (I_{\pi_A} (\mathit{\Delta}))^{-1} \circ A = A. \end{aligned}$$

□

Proof (of Lemma 1, Section 3). By Theorem 1 every general extension is isomorphic to an extension, so it suffices to show that every extension $A : \mathit{\Gamma}^+ \to \mathit{\Gamma}$ is isomorphic to an extension $B : \mathit{\Gamma}^\circ \to \mathit{\Gamma}$ where $\mathit{\Gamma}$ is an initial segment of $\mathit{\Gamma}^\circ$. If $\mathit{\Gamma}$ is already an initial segment of $\mathit{\Gamma}^+$, then we can just take $\mathit{\Gamma}^\circ := \mathit{\Gamma}^+$. Otherwise there exist i and j such that $i < j < s$ and

$$\mathit{\Gamma}^+ = \langle x_0 : \tau_0; \ldots ; x_i : \tau_i; \ldots ; x_j : \tau_j; \ldots ; x_{s-1} : \tau_{s-1} \rangle$$

where x_{i-1} is not in $\mathit{\Gamma}$, $x_i, \ldots, x_{j-1}$ are all in $\mathit{\Gamma}$, and $x_j, \ldots, x_{s-1}$ are all not in $\mathit{\Gamma}$. Because $\mathit{\Gamma}$ is a well formed context, it must be the case that x_{i-1} does not occur in $\tau_i, \ldots, \tau_{j-1}$, and we can safely rearrange $\mathit{\Gamma}^+$ into

$$\begin{aligned} \hat{\mathit{\Gamma}}^+ = \langle x_0 : \tau_0; \ldots ; x_{i-2} : \tau_{i-2}, x_i : \tau_i; \ldots ; x_{j-1} : \tau_{j-1}, \\ x_{i-1} : \tau_{i-1}, x_j : \tau_j; \ldots ; x_{s-1} : \tau_{s-1} \rangle. \end{aligned}$$

This new context $\hat{\mathit{\Gamma}}^+$ is isomorphic to $\mathit{\Gamma}^+$ via $\delta_{\mathit{\Gamma}^+} : \hat{\mathit{\Gamma}}^+ \to \mathit{\Gamma}^+$. Continuing by induction on i, we can eventually construct a $\mathit{\Gamma}^\circ$ which is isomorphic to $\mathit{\Gamma}^+$ via $\delta_{\mathit{\Gamma}^+} : \mathit{\Gamma}^\circ \to \mathit{\Gamma}^+$ such that $\mathit{\Gamma}$ is an initial segment of $\mathit{\Gamma}^\circ$. Therefore $A : \mathit{\Gamma}^+ \to \mathit{\Gamma}$ and $A \circ \delta_{\mathit{\Gamma}^+} : \mathit{\Gamma}^\circ \to \mathit{\Gamma}$ are isomorphic in $\mathbb{E}$. □

Proof. (of Theorem 2, Section 3) To prove that $\mathrm{cod} : \mathbb{E} \to \mathbb{B}$ is a fibration we need to show that for any nominal assignment $u : \mathit{\Gamma} \to \mathit{\Delta}$ from $\mathbb{B}$ and a general extension $A : \mathit{\Delta}^+ \to \mathit{\Delta}$, there exists a Cartesian lifting $\bar{u}(A) : u^* (A) \to A$ in $\mathbb{E}$.

We need to show that $\bar{u}(A)$ is a pullback diagram of the cospan $\mathit{\Gamma} \xrightarrow{u} \mathit{\Delta} \xleftarrow{A} \mathit{\Delta}^+$ in $\mathbb{B}$ and that $u^* (A)$ is a general extension. By Lemma 1 we can assume that A is an extension and $\mathit{\Delta}$ is an initial segment of $\mathit{\Delta}^+$. We can further suppose that the names in $\mathit{\Delta}^+$ are disjoint from the names $\mathit{\Gamma}$, by applying a suitable permutation π to A and noting that if $\mathit{\Delta}$ is an initial segment of $\mathit{\Delta}^+$ then $\pi \cdot \mathit{\Delta}$ is an initial segment of $\pi \cdot \mathit{\Delta}^+$.

Suppose $\mathit{\Delta}^+ = \langle y : \sigma \rangle_0^{m+e-1}$ and $u := \left[y_i \mapsto x_{a(i)} \right]_{i=0}^{m-1}$. Define an extension of $\mathit{\Gamma}$, $\mathit{\Gamma}^+ := \langle x : \tau \rangle_0^{n+e=1}$, where $x_{n+k} := y_{m+k}$ and $\tau_{n+k} := \sigma_{m+k} \left[y_i \mapsto x_{a(i)} \right]_{i=0}^{m-1}$ for all $k < e$. Let $u^* (A) := \delta_{\mathit{\Gamma}} : \mathit{\Gamma}^+ \to \mathit{\Gamma}$. By definition $u^* (A)$ is an extension (and $\mathit{\Gamma}$ is an initial segment of $\mathit{\Gamma}^+$). We can define a nominal assignment $u^+ (A) : \mathit{\Gamma}^+ \to \mathit{\Delta}^+$ by $u^+ (A) := \left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-1} \bowtie \left[y_{m+i} \mapsto x_{n+i}\right]_{i=0}^{e-1}$. By construction, $u \circ u^* (A) = A \circ u^+ (A)$, and so $\bar{u} (A) := \langle u^+, u \rangle : u^* (A) \to A$ is a commutative square.

To show that $\bar{u} (A)$ is a Cartesian lifting of u over A, consider some $X : \mathit{\Xi}^+ \to \mathit{\Xi}$ in $\mathbb{E}$ along with $g : X \to A$ and $v : \mathit{\Xi} \to \mathit{\Gamma}$ such that $u \circ v = \mathrm{cod}\,(g)$. Again, without loss of generality, we can assume that X is an extension and $\mathit{\Xi}$ is an initial segment of $\mathit{\Xi}^+$.

Suppose $\mathit{\Xi} = \langle z : \xi \rangle_0^{r-1}, \mathit{\Xi}^+ = \langle z : \xi \rangle_0^{r+s-1}, v = \left[x_j \mapsto z_{b(j)}\right]_{j=0}^{n-1}, g^+ = \left[y_i \mapsto z_{c(i)}\right]_{i=0}^{m+e-1}$. We can define $v^+ : \mathit{\Xi}^+ \to \mathit{\Gamma}^+$ as $v \bowtie \left[x_j \mapsto z_{c(j)}\right]_{j=m}^{m+e-1}$. To see that v^+ is well formed, recall that $\tau_{n+k} = \sigma_{n+k} \left[y_i \mapsto x_{a(i)}\right]_{i=0}^{m-1}$. Because for all $i < m$, $c\,(i) = b\,(a\,(i))$,

$$\tau_{n+k} \left[v^+\right] = \tau_{n+k} \left[v \bowtie \left[x_j \mapsto z_{c(j)}\right]_{j=m}^{m+e-1}\right] = \sigma_{m+k} \left[y_i \mapsto z_{c(i)}\right]_{i=0}^{m+k-1}$$

which is well typed in $\mathit{\Xi}$ because g^+ is well-defined. Clearly, $u^* (A) \circ v^+ = v \circ X$, and $g^+ = u^+ (A) \circ v^+$ by construction. Therefore, $\bar{v} := \langle v^+, v \rangle : X \to u^* (A)$ is an arrow in $\mathbb{E}$ such that $\bar{u} \circ \bar{v} = g$.

Finally, we need to show that $\bar{v}$ is the unique arrow in $\mathbb{E}$ such that $\bar{u} \circ \bar{v} = g$ and $\mathrm{cod}\,(\bar{v}) = v$. Suppose $\bar{w} : X \to u^* (A)$ is another arrow such that $\bar{u} \circ \bar{w} = g$ and $\mathrm{cod}\,(\bar{w}) = v$. Say $\bar{w}^+ := \left[x_j \mapsto z_{d(j)}\right]_{j=0}^{n+e-1}$. Because $u^* (A) \circ \bar{w}^+ = X \circ v$, it must be the case that for all $i < n$, $d\,(i) = b\,(i)$. Because $u^+ (A) \circ \bar{w}^+ = g^+$, it must be the case that for all $k < e$, $d\,(n+k) = c\,(m+k)$. Therefore, $\bar{w}^+ = \bar{v}^+$, and so $\bar{w} = \bar{v}$. □

$$\begin{tikzcd}
\mathit{\Xi}^+ \arrow[dr, dashed, "v^+"] \arrow[drr, bend left, "g^+"] \arrow[dd, "X"'] & & \\
& \mathit{\Gamma}^+ \arrow[r, "u^+"] \arrow[dd, "u^*(A)"] & \mathit{\Delta}^+ \arrow[dd, "A"] \\
\mathit{\Xi} \arrow[dr, "v"'] & & \\
& \mathit{\Gamma} \arrow[r, "u"'] & \mathit{\Delta}
\end{tikzcd}$$

Proof (of Corollary 2, Section 3). For such a u, a cartesian lifting $\bar{A} (u) : A^* (u) \to u$ is isomorphic to the transpose of $\bar{u} (A)$, and in particular $\bar{u} (A)^+$ will be isomorphic to $A^* (u)$. Since $A^* (u)$ is a general extension, then so is $\bar{u} (A)^+$. □